

Remarks

The present amendment responds to the Official Action mailed November 21, 2000. A petition for a two month extension of the time to respond and authorization to charge Deposit Account No. 50-1058 the fee of \$195 to cover the small entity fee for this extension accompany this amendment. The Official Action required restriction under 35 U.S.C. 121 to one of the following four groups of claims: Group I, claims 1-38 and 49-51; Group II, claims 39-43; Group III, claims 44-48; or Group IV, claims 52-56. Claims 1-38 and 49-51 were rejected under 35 U.S.C. 102(e) based upon Tsushima et al. U.S. Patent No. 6,044,450 (Tsushima). This sole basis for rejection is addressed below following a brief discussion of the present invention to provide context.

Claims 1, 26, and 49 have been amended to be more clear and distinct. Claims 39-48 and 52-56 have been cancelled without prejudice. Claims 1-38 and 49-51 are presently pending.

The Present Invention

The present invention relates generally to improved methods and apparatus for providing abbreviated instructions, mechanisms for translating abbreviated instructions, and configurable processor architectures for system-on-silicon embedded processors and the like. To this end, in accordance with one aspect of the present invention, a program is analyzed and standard length instructions, such as 32-bit instructions, are replaced with abbreviated instructions having a shorter length format, such as 14-bits, tailored to the analyzed program. In this context, translation or conversion means to change from one instruction format into another. As addressed in greater detail below, it does not mean compression as addressed by Tsushima.

The Restriction Requirement

The provisional election made with traverse by telephone is hereby affirmed, and claims 39-48 and 52-56 have now been cancelled without prejudice.

The Art Rejection

As addressed in further detail below, the art rejection is not supported by the relied upon art. By contrast with the present invention, Tsushima principally addresses a compression arrangement for elimination of redundant no operation (NOP) instructions. As stated in the Abstract, each small instruction in a VLIW instruction is added with the number of NOP instructions which succeed the small instruction and these NOP instructions are deleted from the succeeding long instruction. As explained at col. 7, lines 23-53, if there are NOP instructions in the same instruction fields of the succeeding non-compressed VLIW instructions, the number of NOP instructions is stored in the NOP number sub-field of the valid small instruction immediately before the VLIW instructions containing NOP instructions. Thus, if a first instruction 100a as shown in Fig. 2B has a load/store L/S #1 and then seven subsequent VLIW instructions follow it with NOPS in the load/store small instruction field, then NOP number 12 will be 7 and the NOP instruction in the seven subsequent VLIWs will not be stored resulting in a significant compression of the memory required to store the eight VLIWs in our example. This approach of translating native instructions to abbreviated instructions clearly does not correspond to the approach claimed by the present invention and vice versa.

Additionally, according to Tsushima, small instructions in each long instruction are divided into a plurality of groups, and a combination of operation codes of small instructions in each group is replaced by a group code to generate a compressed, grouped instruction. As explained at col. 7, line 54- col. 8, line 14, in the example of Fig. 2C space compression follows

the time compression of Fig. 2B. In place of OP codes of each group, a group code is set which unanimously determines a combination of the OP codes. This group code has a length shorter than the total length of the OP codes. Thus, group code 10A of Fig. 2C is a group code for both L/S #1 (LD/ST #1) and L/S #2 (LD/ST #2). It should be noted that while Tsushima talks of this grouped instruction as compressed "in space", however, with the addition of the NOP number field, it is not clear that the resulting instruction 100b in fact has less bits than the non-compressed VLIW instruction 100.

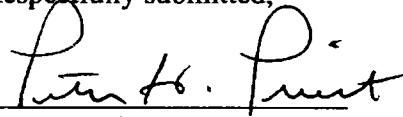
In any case, it should be recognized that Tsushima addresses compression and not abbreviation in the manner presently claimed. Claim 1 has been amended to more clearly and distinctly address this distinction. With respect to claim 18, Tsushima clearly does not determine at least one style pattern of bits that are constant. Tsushima does not fetch "an abbreviated instruction from a memory tailored to a storage of abbreviated instructions having a first fixed number of bits." Further, Tsushima does not address the claimed relationship of SP and PE processing addressed by this claim. With regard to claim 49, Tsushima clearly does not meet this claim as amended.

Given both the complexity of the subject matter of the present invention and the relied upon art, it is believed that a telephone conference interview to include Dr. Pechanek, one of the inventors, may be the most expeditious way to address any issues remaining after the consideration of the present amendment. Should the present rejection be maintained further clarification is requested as to how Tsushima is believed to meet these claims.

Conclusion

All of the claims standing in order for allowance, this case should be promptly allowed. Should there be any issues which might be expedited by a telephone call, the Examiner is requested to call the undersigned at the number below.

Respectfully submitted,



Peter H. Priest
Reg. No. 30,210
Priest & Goldstein, PLLC
529 Dogwood Drive
Chapel Hill, NC 27516
(919) 942-1434

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Specification

Paragraph beginning at page 2, line 4 has been amended as follows:

--In order to meet these opposing requirements, it is necessary to develop a processor architecture and apparatus that can be configured in more [optimum] optimal ways to meet the requirements of the intended task. One prior art approach for configurable processor designs uses field programmable gate array (FPGA) technology to allow software-based processor optimizations of specific functions. A critical problem with this FPGA approach is that standard designs for high performance execution units require ten times the chip area or more to implement in a FPGA than would be utilized in a typical standard application specific integrated circuit (ASIC) design. Rather than use a costly FPGA approach for a configurable processor design, the present invention uses a standard ASIC process to provide software-configurable processor designs optimized for an application. The present invention allows for a dynamically configurable processor for low volume and development evaluations while also allowing optimized configurations to be developed for high volume applications with low cost and low power using a single common architecture and tool set.--

Paragraph beginning at page 3, line 14 has been amended as follows:

--In one embodiment of the present invention, a manifold array (ManArray) architecture is adapted to employ various aspects of the present invention to solve the problem of configurable application-specific instruction set optimization and program size reduction, thereby increasing code density and making the general ManArray architecture even more desirable for high-volume and portable battery-powered [type] types of products. The present invention extends the pluggable instruction set capability of the ManArray architecture described

in U.S. Application Serial No. 09/228,374 filed December 18, 1998, entitled "Methods and Apparatus for Scalable Instruction Set Architecture with Dynamic Compact Instructions" with new approaches to program code reduction and stand-alone operation using only abbreviated instructions in a manner not previously described.--

In the Claims

Claims 39-48 and 52-56 have been cancelled without prejudice.

Claims 1, 26 and 49 have been amended as follows:

1. (amended) A method for generating an application specific program utilizing an abbreviated instruction set comprising the steps of:

generating a native program for an application utilizing a set of native instructions having a first fixed number of bits;

debugging the native program;

processing the debugged native program to determine an abbreviated instruction set having a second fixed number of bits less than the first fixed number of bits and corresponding to the set of native instructions; and

converting the native program to the application specific program by replacing the set of native instructions with the abbreviated instruction set.

26. (amended) A method for translating abbreviated instructions into a native instruction format comprising the steps of:

fetching an abbreviated instruction having a first fixed number of bits from a memory tailored to storage of abbreviated instructions;

dynamically translating the abbreviated instruction into the format of a native instruction in a sequence processor (SP) array controller said native instruction having a second fixed number of bits greater than said first fixed number; and

dispatching the native instruction to a processing element for execution.

49. (amended) A system for translating abbreviated instructions into a native instruction format comprising:

a memory storing an abbreviated instruction having a first fixed number of bits;

means for fetching the abbreviated instruction from the memory; and

means for dynamically translating the abbreviated instruction into a native instruction in the native instruction format having a second fixed number of bits greater than said first fixed number.